Two Formal Systems for Proving Assertions About Programs (プログラムの基礎理論)

AUTHOR(S):

IGARASHI, SHIGERU

CITATION:

IGARASHI, SHIGERU. Two Formal Systems for Proving Assertions About Programs (プログラムの基礎理論). 数理解析研究所講究録 1973, 189: 2-6

ISSUE DATE:

1973-10

URL:

http://hdl.handle.net/2433/107226

数理解析研究所講究録
189巻 1973年 2-6

TWO FORMAL SYSTEMS FOR PROVING ASSERTIONS ABOUT PROGRAMS

Shigeru Igarashi (Kyoto University)

1. First-order logic of typed theories.

1.1 Types. α, β, γ .... denote types. Ordered types are denoted by αo, βo, etc.
a) We presuppose that there are finite number of base types.
b) α, β are types -----→ $\alpha\to\beta$ is a type.
c) $\alpha o \to \beta o$ is a type -----→ $(\alpha o \to \beta o)o$ is a type.

1.2 Alphabet.
α-constants
α-variables (for each α)

$(\alpha 1, \ldots, \alpha n)$-predicates
logical symbols:

(,) Min = $\vee$ $\exists$ $\neg$

1.3 Terms.
a) a is an α-constant -----→ a is an α-term.
b) x is an α-variable -----→ x is an α-term.
c) t is an $\alpha\to\beta$-term, u is an α-term -----→ t(u) is a β-term.
d) t is an $(\alpha o \to \alpha o)$-term -----→ Min t is an αo-term.

1.4 Interpretation.
Definitions.
$\aleph_0$-inductively ordered set. L is nonempty. Any linearly ordered subset (nonempty) X of L has sup X in L. (countable)

f: L → L' is continuous iff

$$f(\sup X) = \sup f(X) \qquad (1)$$

for any monotone increasing sequence X in L.

a) α is a base type that is not ordered -----→ $D\alpha$ is a nonempty set as the domain of individuals.
b) αo is an ordered base type -----→ $D\,\alpha o$ is an 0-inductively ordered set with the least element O.
c) $D(\alpha\to\beta)$ is the set of functions of $D\alpha$ into $D\beta$.
d) $D(\alpha o \to \beta o)o = \{ f \mid f: \text{continuous}, f \in D(\alpha^0 \to \beta^0) \}$.

t(u) denotes the application of t to u.

$$\text{Min } f = \sup\{ f(O), ff(O), fff(O), \ldots \} . \qquad (2)$$

Logical axioms.

propositional axiom. $A \vee A$.

identity axiom. $x=x$.

equality axiom. $x=y \to \text{Min } x = \text{Min } y$.
$x=y \to z(x) = z(y)$.
$x1=y1 \to \ldots \to xn=yn \to p(x1, \ldots, xn) \to p(y1, \ldots, yn)$.

extensionality axiom. $x=y \equiv \forall z(x(z) = y(z))$.

stationariness axiom. $x(\text{Min } x) = \text{Min } x$.

induction axiom (fixed-point induction).

$$A[0] \to \forall y(A[y] \to A[x(y)]) \to A[\text{Min } x].$$

2. Admissibility of fixed-point induction.

Truth functions are functions into the two element complete lattice.

2.1 Hierarchy of admissibility.

I. a.i.w.
II. a.i.s. $(f(\sup X)=\text{limsup } f(X))$
III. weakly continuous. $(f(\sup X)=\text{liminf } f(X) = \text{limsup } f(X))$
IV. continuous.
V. constant.

2.2 Inheritance tables.

$A \vee B$

A	B	a.i.w.	a.i.s.	w.cont.	const.
a.i.w.		x*)	x	x	x
a.i.s.		x	a.i.s.	a.i.s.	a.i.s.
w.cont.		x	a.i.s.	w.cont.	w.cont.
const.		x	a.i.s.	w.cont.	const.

*) becomes a.i.w. in case of conjunction.

2.3 Elementary formulas.
Theorems. Scott's awffs of the form $t \leq u$ are a.i.s.
If D α o is discrete (upward) (No ascending chains that interpolate two elements of D α o exist.), then Scott's awffs of type α o are weakly continuous.
For A to be weakly continuous it is necessary and sufficient that A and $\neg A$ are a.i.s. (ETC.)

3. Formal system representing assertions for ALGOL-like statements.**)

3.1 Statements.

a) q is an (m,n)ary procedure symbol,
xl, ... ,xm are variables,
tl, ... , tn are terms in L(T)

---------- → q(xl, ... ,xm;tl, ... ,tn) is an atomic statement.

b) A, B are statements -----→ A;B is a statement.

c) A, B are statements, F is a quantifier-free formula in L(T)
---------- → if F then A else B is a statement.

3.2 Assertions(wffs).

i) F { A } G. F, G are formulas in L(T), A is a statement.
ii) p(xl, ... ,xm;yl, ... ,yn) proc A.
iii) Formulas in L(T).

3.3 Axioms.

primitive procedures.

assignment axiom. $R(f)\{ x \leftarrow f \} R(x)$.

invariant axiom. $R\{ q(xl, \ldots ,xm;tl, \ldots ,tn) \} R$.

xl, ... , xm do not occur free in R.

defining axioms for procedures. Wffs of the form (ii) of 3.2.

logical axioms. Theorems belonging to the theory T.

3.4 Inference rules.

logical rules.

$$(1)\quad \frac{P \rightarrow Q \qquad Q\{ A \}R}{P\{ A\} R}\ . \qquad (2)\quad \frac{P \{A\} R \qquad R \rightarrow S}{P\{ A\} S}\ .$$

$$(3)\quad \frac{P \{A \}R \qquad Q \{A \}S}{P \overset{\vee}{\wedge} Q \{ A \} R \overset{\vee}{\wedge} S}\ .$$

**) This is an exposition of the study by London, Luckham, and Igarashi.

substitution rules.
$$\frac{P(x)\ \{q(x;t(x))\}\ R(x)}{P(z)\ \{q(z;t(z))\}\ R(z).} \quad (4)$$

z denotes distinct variables which do not occur free in P(x), t(x), or in R(x).

$$\frac{P(y)\ \{q(x;t(y))\}\ R(y)}{P(u)\ \{q(x;t(u))\}\ R(u).} \quad (5)$$

x does not occur free in u.

recursion rule.
$$\frac{[\ P\ \{q(x;y)\}\ R\ ]}{P\ \{K(q)\}\ R \qquad r(x;y)\ \underline{\text{proc}}\ K(r)}{P\ \{r(x;y)\}\ R}. \quad (6)$$

q is a free procedure variable that does not occur free in any of the upper formulas except those places that are explicitly so indicated.

rules for constructors.
$$\frac{P\{A\}\ Q \qquad Q\ \{B\}\ R}{P\{A;B\}\ R}. \quad (7)$$

$$\frac{P\&F\{A\}\ R \qquad P\&\neg F\ \{B\}\ R}{P\{\text{if } F \text{ then } A \text{ else } B\}R.} \quad (8)$$

3.5 Relatively sound rules.
$$\frac{P\&F\{A\}P}{P\{\text{while } F \text{ do } A\}\ P\&\neg F.} \quad (9)$$

The following rule is a derived rule relative to (9).
$$\frac{P\&F\ \{A\}\ P \qquad P \xrightarrow[\&\neg F]{} Q}{P\{\text{while } F \text{ do } A\}\ Q}. \quad (10)$$

3.6 "Verification conditions"

A sufficient set of formulas in L(T) to prove $P\{A\}R$ is called a set of verification conditions for $P\{A\}R$. There is an algorithms to get this set from any given goal to be proved, which is a kind of backward derivation and similar to parsing procedures. A practical version of this algorithm has been implemented for PDP-10 of the Artificial Intelligence Project, Stanford University, by London, and has turned out to be extremely useful.

E.g., $AH(x\ f, R) = Subst(R, x, f)$.

$AH(\text{if } F \text{ then } A \text{ else } B, R) = (F \rightarrow AH(A, R)) \& (\neg F \rightarrow AH(B, R))$.

$AH(A;B, R) =$ } left to the reader.

$AH(\text{while } F \text{ do } B, R) =$ } left to the reader.

$AH(q(z;t), R) = Pre(q) \& \forall x(Subst(Res(q), y, t) \rightarrow R)$.

(Cf. the rule of adaptation (Hoare))

$VC(P, A, R) = P \rightarrow AH(A, R)$.

3.7 Consistency and strengthening the interpretation for proving termination.

These problems are being successfully studied. We have a consistency proof up to the recursion rule, and also a formal system for proving strong correctness (involving termination).